2010 International Conference on Measuring Technology and Mechatronics Automation

A Low- cost Multi-standard Audio Decoder Architecture Design

Tao Zhang, Fengping Yu, Haojun Quan
Electronic and Information Engineering School, TJU
Tianjin, China
e-mail:zhangtao@tju.edu.cn

***Abstract*—At present, audio decoder in multimedia processing chip must support multi-standard, such as AC-3, MP3, WMA. Many methods, from software to hardware can be used to implement the architecture. In this paper, an architecture based on Embedded Processor core is proposed to support multi-standard audio decoding. Embedded Processor core is used for real-time audio decoding and rendering. Featured with preload method, multi-standard audio decoder could be extended easily. Further more, data and instruction memory could be updated during Embedded Processor core works, so on-chip data and instruction memory could be reduced largely, which means very low cost. As we can see from the experiment result, audio decoder based on this architecture could support current popular audio standards real-time at very low cost.**

Keywords-Low cost; Multi-standard;Audio Decoder;Preload

I. INTRODUCTION

At present, Audio Decoder in multimedia processing chip must support multi-standards, such as MPEG-1/2 layer1-3[1][2], MPEG AAC, Dolby AC-3[3], China AVS[4], Microsoft WMA. They are widely used in portable players, cell phone and digital TV.

Many implementation methods of audio decoder are available, such as full hardware decoder, embedded (DSP, ARM or MIPS) software decoder and software decoder with hardware accelerator. For the full hardware decoder [5], because the circuit is fixed, it could be optimized for certain application, so the cost is the lowest. But it is difficult to modify or update it in order to extend new functions. And the optimal design for the common modules in different standards is very time consuming and is dependent on the understanding to algorithms. The same problem exists in software decoder with hardware accelerator. For the software decoder[6][7], it is very easy to update or modify the decoder function. But in order to support multiple standards at the same time, large on-chip memory is required. If the extern memory is used, the decoding speed is very slow.

Based on the embedded software decoder method, this paper focuses on reducing on-chip memory requirement. For different audio decoding standard, and for different decoding stage in one standard, data and instruction preload mechanism are applied. To accelerate the decoding processing, preload mechanism is improved with ping-pang buffer. As an example, this audio decoder architecture is configured based on a DSP core. With small size on-chip RAM, MPEG audio, Dolby AC-3 and China AVS audio decoder are implemented.

II. LOW-COST AUDIO DECODER ARCHITECTURE DESIGN

A. *Architecture Design*

The system block diagram is shown in figure 1.

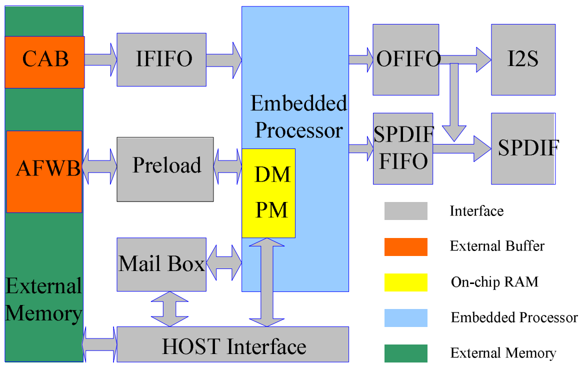

Figure 1. The system block diagram.

Similar to an ordinary audio decoder, in system level a large buffer, which is called CAB (Coded Audio Buffer), is allocated in external memory, for example DDR SDRAM. The Embedded Processor fetches input data from CAB through IFIFO interface. In the Embedded Processor module, on-chip DM (Data Memory) is the data RAM using to store the data, on-chip PM (Program Memory) is the instruction RAM using to store the instructions. Both of them can be initialized through the HOST Interface. Considering the efficiency, all instruction and data required by the decoding task should be in on-chip RAM before running. The decoded PCM data is stored in OFIFO. PCM data is packed into I2S format[8] through I2S interface and then output. The S/PDIF output[9] is also supported, both linear PCM[9] and non-linear PCM[10]. Linear PCM data is output from OFIFO directly, and non-linear PCM is output by Embedded Processor through SPDIF FIFO.

The decoding is a sequential processing, and only the running instruction and related data is in on-chip memory, the decoding efficiency is guaranteed. So in order to reduce on-chip memory cost while support multi-standard audio decoder, the whole large instruction and data space, which is required by multi-standard audio decoder, could be divided into many small sections. Data and instruction could be loaded into on-chip RAM before running. So it is called “preload”. In embedded system, the software running on embedded processor is named Firmware. In this architecture, a large size buffer is allocated in external memory to store the whole firmware. This is AFWB (Audio FirmWare Buffer) in Fig.1. AFWB consists of many small firmware sections, which could be preload into Embedded Processor DM/PM before running by Preload Module.

Mailboxes are used for communication between the Embedded Processor and Host CPU, for example configure decoder operation and get decoder status.

B. *Principle of Operation*

- Preload and post-save mechanism

As described before, the whole audio decoder firmware is divided into small section, clustered as different standard decoder. As illustrated in Fig. 2 (a).

On-chip memory consists of two parts: fixed DM/PM and preloaded DM/PM, which is shown in Fig. 2(b). When the system is started, the host CPU first loaded the common firmware into on-chip memory, fixed DM/PM, and all of other firmware into AFWB which is in external Memory for preload. Then CPU enables the DSP core and begin to transfer the audio stream from extern to CAB. The common firmware in the on-chip memory will get the audio stream from CAB via IFIFO. Then the stream is parsed and audio type is determined, corresponding audio firmware is known. After that it will load the required firmware from AFWB to on-chip memory PM, DM, with preload module.

Once the start preload process is over, the Embedded Processor core program counter (PC) will jump to the preloaded firmware entry and the firmware starts to decode the audio stream. In the decoding process, there’s preload also. When the running firmware encounters data or instruction out of range of on-chip memory, new preload operation is activated.

In order to reduce on-chip memory size, the fixed memory size is small. So the intermediate data in decoding couldn’t be saved on-chip. Before new preload start, some intermediate data must be transferred into AFWB. This operation is named as post-save. With preload and post-save mechanism, the decoding procedure could be done.

After decoding a frame, the DSP core will send the PCM samples to the OFIFO and the I2S interface.

- Accelerating with Ping-pang Structure

Audio decoder architecture with preload and post-save mechanism reduces on-chip memory cost significantly. However, because PC must be halted before preload and post-save operation complete, the decoding speed is very slow. AFWB is allocated in external large memory, for example SDRAM, FLASH ROM. Compared with on-chip memory, the access speed of FLASH ROM is very slow. Even the access speed of SDRAM is high enough, but arbitrator is required for SDRAM, the access efficiency is lower.

To improve decoding speed, preloaded memory is doubled with ping-pang structure, as illustrated in Fig.2 (c). Here when firmware is running in ping preloaded memory, preload module transfers (preload or post-save) pang preloaded memory content; and when firmware is running in pang preloaded memory, ping preload memory is transferred. In system level design, timing is guaranteed that preload memory transfer time is less than running time for ping and pang preload memory.

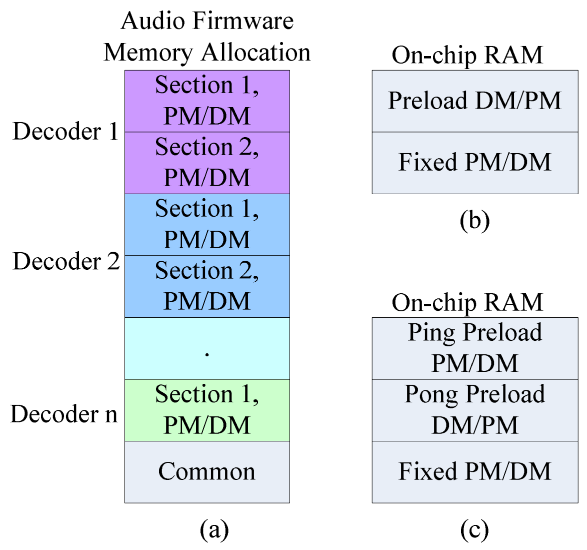

Figure 2. Memory allocation for audio decoder

III. Low-cost Audio Decoder Architecture Implementation

A. *Embedded Processor*

The Embedded Processor is the core of the decoder. Here a 16bit fixed-point Digital Signal Processor is selected, which could provide 100MIPS processing performance at 100MHz clock . The architecture of the DSP core is shown in Figure 3.

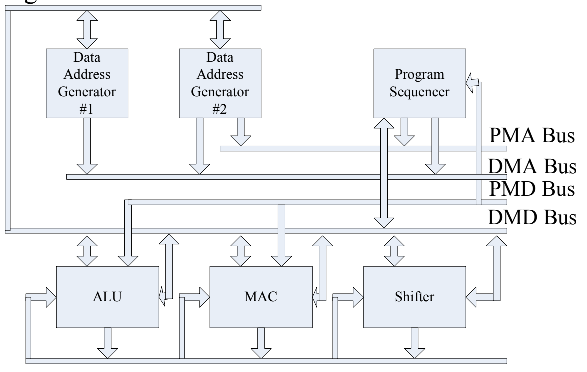

Figure 3. DSP Core architecture

The DSP core contains three independent, full-function computational units: an arithmetic/logic unit (ALU), a multiplier/accumulator (MAC) and a barrel shifter. The computation units process 16-bit data directly and provide hardware support for multi-precision computation as well.

Two dedicated data address generators and a powerful program sequencer ensures efficient use of the computational units. The data address generators (DAGs) provide memory addresses when memory data is transferred to or from the input or output registers. Each DAG keeps track of up to four address pointers. When a pointer is used for indirect addressing, it is post-modified by a value in a specified register. With two independent DAGs, the processor can generate two addresses simultaneously for dual operand fetches.

The processors have five internal buses:

Program Memory Address (PMA) and Data Memory Address (DMA) buses— Used internally for the addresses associated with Program and Data Memory.

Program Memory Data (PMD) and Data Memory Data (DMD) buses — Used for the data associated with memory spaces. These buses are multiplexed into a single external address bus and a single external data bus; the BMS, DMS and PMS signals select the different address spaces.

Result (R) bus—Transfers intermediate results directly between the various computational units.

The PMA bus is 14 bits wide allowing direct access of up to 16 K words of mixed instruction code and data. The PMD bus is 24 bits wide to accommodate the 24-bit instruction width. The DMA bus is 14 bits wide allowing direct access of up to 16 K words of data. The DMD bus is 16 bits wide. The DMD bus provides a path for the contents of any register in the processor to be transferred to any other register or to any Data Memory location in a single cycle. The Data Memory address comes from two sources: an absolute value specified in the instruction code (direct addressing) or the output of a data address generator (indirect addressing). Only indirect addressing is supported for data fetches from Program Memory. The PMD bus can also be used to transfer data to and from the computational units through direct paths or via the PMD-DMD bus exchange unit. The PMD-DMD bus exchange unit permits data to be passed from one bus to the other. It contains hardware to overcome the 8-bit width discrepancy between the two buses, when necessary.

B. *MPEG Audio Decoder Implementation Example*

Take MPEG audio decoder for example, including MPEG1/2 layer 1-3. Because the on-chip memory is not enough, preload is necessary.

In MPEG audio decoder, the Huffman and Stereo parts of PM are preloaded to the on-chip PM if needed. Besides the Huffman part need about 7K bytes for Huffman tables which is optimized already and the Subband synthesis part need 8K bytes to store the middle result of synthesis buffer and moreover it is used in current frame and must be updated to be used in the next frame, so the 8K PDM are used for them. Before the Huffman part, the firmware preloads the Huffman tables from AFWB to PDM and preloads the synthesis buffer from AFWB to PDM before Subband synthesis. Because the data in the synthesis buffer is needed when decoding the next frame, it must be preloaded from PDM to AFWB for the next frame.

As for DM, because a lot of tables are required the size of them is 12K bytes; these tables are placed into AFWB at first and preloaded to specified DM area when they are need. Because there’s a selection of tables for each frame, so only one of them need be preloaded, whose largest size is 4K bytes; 2K bytes are required in the Subband synthesis part to store the window coefficients. Therefore 4K bytes space in DM is reserved for them. And because the tables and window coefficients are read only, they are just preloaded into the reserved area when needed.

Experiment Result

Using the Synopsys DC 2004.6 and the on chip memory is generated by the SMIC library. According to the synthesis report, the DSP core occupies 80,000 gates and the on-chip RAM occupies less than 350,000 gates, other logic modules use about 100,000 gates. The total gate count used is 512,347.

A low-cost multi-standard audio decoder is implemented based on the system. At present three standard families, MPEG1/2 layer1-3, AC-3 and AVS audio are supported. The experiment result is illustrated in the Table 1.Compared with the result in table 2, which is referenced from Tensilica Audio Specific DSP HiFi2 engine [11], this audio decoder is very low-cost.

TABLE I. ON-CHIP MEMORY REQUIREMENT FOR AUDIO DECODER (KB)

	PM	DM	PDM(ROM)
MPEG	13	10	8
AC-3	10	8.5	4
AVS*	13	12	8

TABLE II. ON-CHIP MEMORY REQUIREMENT FOR AUDIO DECODER BASED ON HIFI2 ENGINE (KB).

	P M	D M	R OM
MPEG[12]	20 k	2 8k	1 5k
AC- 3[13]	20 k	2 7k	8 k

At first, because AVS audio is a new standard developed by China Audio and Video coding Standard group, it is not support on Tensilica HiFi2 engine now. So there no compare.

Secondly, if required, the on-chip memory could be reduced further in this architecture, if the preload frequency is increased. But this will produce too small data/instruction section, and the preload frequency increases more than linear speed to section number, because more preload and post-save are required if section is too small.

At last, in this example, the memory allocation is background with balance between on-chip DM and PM size, we intend to do it. To reduce on-chip memory, reducing DM further is suggested because few post-save is introduced.

IV. CONCLUSION

Nowadays multi-standard audio decoder must be supported in multimedia applications. Implementations based on software and/or hardware are all available. In this paper, based on embedded processor and preload mechanism, a low cost architecture is proposed. Featured with its ping-pang preload method, its function could be extended easily. As we can see from the experiment result, requirement of current popular audio standards could be meet at very low memory cost.

REFERENCES

[1] ISO/IEC11172-3, Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s, Part 3: Audio, MPEG-1, 1992.

[2] ISO/IEC 13818-3, Information Technology—Generic Coding of Moving Pictures and Associated Audio, Part 3: Audio, MPEG-2, 1994.

[3] Digital Audio Compression Standard (AC-3, E-AC-3) Revision B, Document A/52B of Advanced Television System Committee (ATSC), 14 June 2005

[4] Information Technology—Audio and Video coding Standard, Part 3 Audio, April, 2006

[5] Kyoung Ho Bang, Nam Hun Jeong2, Joon Seok Kim2, Design and VLSI Implementation of a Digital Audio-specific DSP Core for MP3/AAC, IEEE Trans. Consumer Electronics, Vol. 48, No. 3, Aug 2002.

[6] Steve Vernon, Design and Implementation of AC-3 coders, IEEE Transactions on Consumer Electronics, Vol.41, No. 3. 754-759

[7] Lee, K.S., Park, Y.C., and Youn, D.H., Software Optimization of the MPEG-audio Decoder using a 32-bit MCU RISC Processor, IEEE Transactions on Consumer Electronics, 2002, 48, pp. 671–676

[8]Philips, Inter IC Sound Bus, http://www.semiconductors.philips.com/acrobat/various/I2SBUS.pdf, June, 1996

[9] IEC 60958, Digital audio Interface, 2006.05.23

[10] IEC 61937, Digital audio – Interface for non-linear PCM encoded audio bit streams applying IEC 60958. 2006.05.23

[11] Steve Leibson, Designing SOCs with Configured Cores[M], Morgan Kaufmann Publishers, 2006

[12] http://www.tensilica.com/products/audio/audio-codecs---am3d/mp3-decode.htm

[13] http://www.tensilica.com/products/audio/audio-codecs---am3d/dolby-digital-ac-3-decode-5-1-channel.htm